REMARKS

Applicant is in receipt of the Office Action mailed March 24, 2004. Claims 1-15 are pending in the application. Applicant has cancelled 1-12. Furthermore, Applicant has added new claims 16-24 that substantially contain the subject matter of claims 1-12 to more fully characterize the claimed invention. Applicant believes that these new claims are allowable as currently written. Further consideration of the present case is earnestly requested in light of the following remarks.

§ 112 Rejections

Claims 1-12 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Specifically, the Examiner points to the use of the word "characterized" as being inappropriate.

Applicant has cancelled claims 1-12.

§ 103 Rejections

Claims 1, 3, 5, 10-15 were rejected under 35 U.S.C. 103(a) as being unpatentable over Jensen et al (U.S. Patent No. 5,293,597) in view of Allegrucci et al (U.S. Patent No. 5,428,779) and in further view of Baker et al (U.S. Patent No. 5,369,749). Claim 2 is rejected under 35 U.S.C. 103(a) as being unpatentable over Jensen in view of Allegrucci and in further view of Golson (U.S. Patent No. 5,390,332). Claim 4 is rejected under 103 U.S.C. 103(a) as being unpatentable over Jensen in view of Allegrucci and in further view of Devic (U.S. Patent No. 5,987,582). Claim 7 is rejected under 35 U.S.C. 103(a) as being unpatentable over Jensen in view of Allegrucci and in further view of Kalaynaraman (Process Management Concepts). Claim 8 is rejected under 35 U.S.C. 103(a) as being unpatentable over Jensen in view of Allegrucci and in further view of Endicott (U.S. Patent No. 6,029,206). Claim 9 is rejected under 35 U.S.C. 103(a) as being unpatentable over Jensen in view of Allegrucci and in further view of NEC (Server-Dispensing Database Implementation Procedure). Applicant respectfully traverses these rejections based on the following reasoning.

As held by the U.S. Court of Appeals for the Federal Circuit in Ecolochem Inc. v. Southern California Edison Co., an obviousness claim that lacks evidence of a suggestion or motivation for one of skill in the art to combine prior art references to produce the claimed invention is defective as hindsight analysis.

In addition, the showing of a suggestion, teaching, or motivation to combine prior teachings "must be clear and particular . . .. Broad conclusory statements regarding the teaching of multiple references, standing alone, are not 'evidence'." *In re Dembiczak*, 175 F.3d 994, 50 USPQ2d 1614 (Fed. Cir. 1999). The art must fairly teach or suggest to one to make the specific combination as claimed. That one achieves an improved result by making such a combination is no more than hindsight without an initial suggestion to make the combination.

<u>Claim 13</u>

Claim 13 of the Application recites:

> "A method for the direct call of a target function by means of a start function in a computer system comprising a processor with a memory management unit (MMU), wherein the computer system includes an operating system, wherein the start function is a component of a first task with a first memory context and the target function is in another memory context, the method comprising:
> the first task executing the start function to perform a context switch from the first memory context into the other memory context;
> executing the target function in the other memory context;
> reversing said context switch to return to the first memory context after executing the target function."

Jensen teaches a concurrent context memory management unit. Jensen discloses <u>three processes</u> including a process A copying data from a process B to process C. Jensen does not teach a start function that is a component of a first task with a first memory context and a target function that is in another memory context, where the target function is directly called by means of a start function. Instead, Jensen teaches interprocess data transfers that "eliminates the need to construct temporary mapping tables when performing the data transfer operation." (Abstract) Furthermore, Jensen teaches that "in FIG. 2, <u>there are three processes</u>, 201, 202 and 203 (<u>A, B and C</u>). Process

A is a privileged process, and processes B and C are normal (unprivileged) processes. Processes B and C request that process A copy a block of data from process B (L bytes of data starting at virtual address VB) to process C (starting at address VC)" (*Emphasis Added*) (Col. 3, lines 50-56.)

Jensen teaches away from a start function that is a component of a first task with a first memory context and a target function that is in another memory context, where the target function is directly called by means of a start function. Instead, Jensen teaches copying and transferring data between three processes, which is significantly different from a direct call of a target function by a start function by means of a processor as cited in claim 13.

Allegrucci teaches a hardware implementation and a method of operation of an automatic context switch system. Specifically, Allegrucci teaches a context switching system and method for saving, restoring, and switching tasks. However, Allegrucci does not teach or suggest a processor with a memory management unit (MMU) that has a start function which is a component of a first task with a first memory context and a target function which is a second component of a second task with a second memory context.

Instead Allegrucci teaches a hardware implementation of a context switching system, as describes in column 2, line 66 – column 3, line 2:

> "The present invention is directed to a hardware implementation of an automatic context switch mechanism (the system) and a method of operation of the same."

Allegrucci does not teach a direct call of a target function by means of a start function, but instead teaches a hardware implementation of context switching. According to Allegrucci, the system uses a hardware context queue to perform the switching:

> "In a preferred embodiment of the present invention, the instructions are loaded in a temporary program queue inside the DMA. The DMA executes local instructions, or passes other instructions and data to the functional blocks for processing." (col. 4 lines 13-17)

Allegrucci does not teach nor suggests a start function that is a component of a first task with a first memory context and a target function that is in another memory context, where the target function is directly called by means of a start function. Instead, Allegrucci teaches a method and a system for context switching for saving, restoring, and switching tasks, which is significantly different from a direct call of a target function by a start function by means of a processor as cited in claim 13.

Baker teaches a method and apparatus for the direct transfer of information between application programs running on distinct processors without utilizing the services of one or both operating systems. Specifically, the Examiner points to col. 28 lines 60-68 to col. 29 lines 1-33, col. 21, line 54 through col. 22, line 5, and col. 15, line 55 through col. 16, line 10 as disclosing a direct call of a target function by a start function. Applicant respectfully disagrees.

Specifically, in col. 28 line 60 – col. 29 line 33 Baker describes the elements of Figure 10:

> "Thus FIG. 10 is used to illustrate a preferred form of the mapping of the S/88 virtual storage to real storage 16 by a storage management unit 105 for one module 9. The virtual address space 106 is divided into S/88 operating system space 107 and user application space 108. Within the space 107 is an area 109 (addresses 007E0000 to 007EFFFF) reserved for hardware and code used to couple each S/370 processor element to a respective S/88 processor element in a processor unit such as 21. The address space 109 is made transparent to the S/88 operating system during normal system processing. The use of this space 109 will be described in detail below.
>
> During system initialization, the storage management unit 105 assigns within the S/88 main storage unit 16 a S/370 main storage area for each set of four S/370 processor elements in partnered units such as 21 and 23. Thus three S/370 main storage areas 162, 163 and 164 are provided for partner units 21, 23 and 25, 27 and 29, 31 respectively. The S/88 processor elements within the partner units access the remaining parts of the storage unit 16 in the manner described in the Reid patent.
>
> The S/370 storage areas 162-164 are assigned, as will be described later, in a manner such that the S/88 operating system does not know that these areas have been "stolen" and are not reassignable to S/88 users by the storage management unit unless returned to the S/88 space. Since the S/370 systems are virtual systems, they access their respective main storage area via address translation. The partner S/88 main storage unit 18 requires identical S/370 main storage areas (not shown). Each S/370 processor element can access only its respective S/370 main storage area and produces an error signal if it attempts to access the S/88 main storage space. Each S/88 processor element, however, can access (or direct

the access to) the S/370 main storage area of its respective S/370 processor element during S/370 I/O operations when the S/88 processor element acts as an I/O controller for its S/370 processor element."

In other words, in the passage above Baker describes a preferred form of the mapping of the S/88 virtual storage to real storage 16 by a storage management unit 105 for one module 9 as illustrated in Figure 10 of Baker. Baker does not teach or suggest a start function that is a component of a first task with a first memory context and a target function that is in another memory context, where the target function is directly called by means of a start function. Instead, Baker teaches mapping of the S/88 virtual storage to real storage by a storage management unit for one module, which is significantly different from a direct call of a target function by a start function by means of a processor as cited in claim 13.

Furthermore, in col. 21, lines 54 – col. 22, line 5 Baker summarizes his invention as follows:

"6. Summary

Thus, the functions of two virtual operating systems (e.g., S/370 VM, VSE or IX370 and S/88 OS) are merged into one physical system. The S/88 processor runs the S/88 OS and handles the fault tolerant aspects of the system. At the same time, one or more S/370 processors are plugged into the S/88 rack and are allocated by the S/88 OS anywhere from 1 to 16 megabytes of contiguous memory per S/370 processor. Each S/370 virtual operating system thinks its memory allocation starts at address 0 and it manages its memory through normal S/370 dynamic memory allocation and paging techniques. The S/370 is limit checked to prevent the S/370 from accessing S/88 memory space. The S/88 must access the S/370 address space since the S/88 must move I/O data into the S/370 I/O buffers. The S/88 Operating System is the master over all system hardware and I/O devices. The peer processor pairs execute their respective Operating Systems in a single system environment without significant rewriting of either operating system."

In other words, in the above passage Baker teaches advantages of combining two virtual operating systems into a single system without significant rewriting of either operating system. Baker does not teach or suggest a start function that is a component of a first task with a first memory context and a target function that is in another memory context, where the target function is directly called by means of a start function. Instead, Baker teaches combining two virtual operating systems into a single system without significant rewriting of either operating system, which is significantly different from a direct call of a target function by a start function by means of a processor as cited in claim 13.

Lastly, Examiner quotes Baker col. 15, line 55 through col. 16, line 10, which describes a feature of sharing a real storage between two or more processors that are executing different virtual storage operating systems:

> "4. Sharing a Real Storage Between Two or More Processors Executing Different Virtual Storage Operating Systems
>
> <u>This feature couples a fault tolerant system to an alien processor and operating system that does not have code to support a fault tolerant storage, i.e. code to support removal and insertion of storage boards via hot plugging, instantaneous detection of corrupted data and its recovery if appropriate, etc.</u>
>
> <u>This feature provides a method and means whereby two or more processors each executing different virtual operating systems can be made to share a single real storage in a manner transparent to both operating systems, and wherein one processor can access the storage space of the other processor so that data transfers between these multiple processors can occur</u>.
>
> This feature combines two user-apparent operating systems environments to give the appearance to the user of a single operating system. Each operating system is a virtual operating system that normally controls its own complete real storage space. This invention has only one real storage space that is shared by both processors via a common system bus. Neither operating system is substantially rewritten and neither operating system knows the other exists, or that the real storage is shared.

In other words, in the above passage Baker describes benefits and use of the feature of sharing a real storage between two or more processors that are executing different virtual storage operating systems. Baker does not disclose a start function that is a component of a first task with a first memory context and a target function that is in another memory context, where the target function is directly called by means of a start function. Instead, Baker teaches sharing memory between two or more processors, which is significantly different from a direct call of a target function by a start function by means of a processor as cited in claim 13.

In conclusion, Jensen, Allegrucci, and Baker, taken singly or in combination, do not teach or suggest a start function that is a component of a first task with a first memory context and a target function that is in another memory context, where the target function is directly called by means of a start function. Furthermore the Applicant submits that neither Jensen, Allegrucci, and Baker provide a motivation to combine.

Similar arguments apply with equal force to independent claims 16 and 24. Applicant thus respectfully submits that each of the independent claims 13, 16, and 24, and claims dependent thereon, are patentably distinct over the cited art, and are thus allowable. Thus, Applicant respectfully requests that the section 103 rejection of claims 13-15 be removed. Due to similarity of claims 16-24 to cancelled claims 1-12, Applicant submits that claims 16-24 are allowable.

Applicant also asserts that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

Applicant submits the application is in condition for allowance, and an early notice to that effect is requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert & Goetzel PC Deposit Account No. 50-1505/5150-45700/JCH.

Also enclosed herewith are the following items:

☒ Return Receipt Postcard

Respectfully submitted,

Mark S. Williams

Mark S. Williams
Reg. No. 50,658
AGENT FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel PC
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800
Date: June 24, 2004